บทความพิเศษ

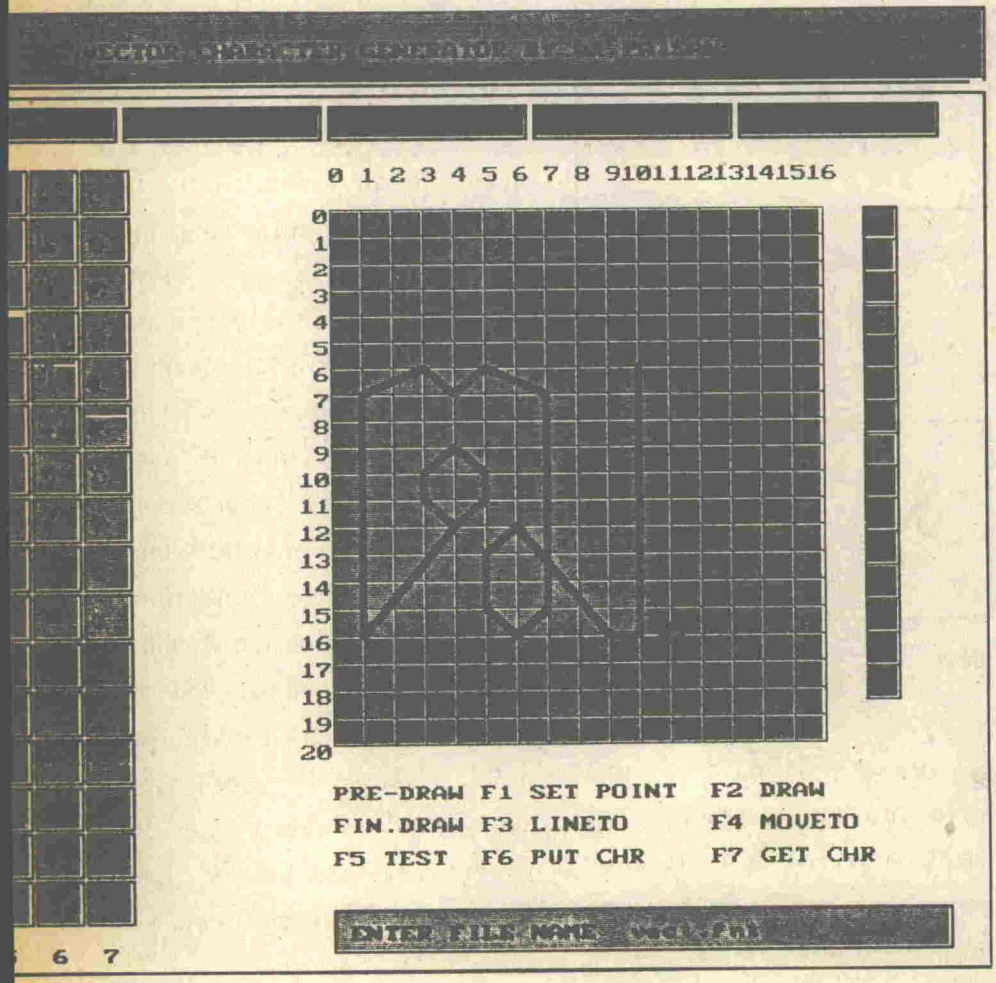
FONT (THAI)

ดร.ไพศาล สงวนหมู่

การสร้างฟอนต์
เพื่อกราฟิกแบบเวคเตอร์

SOFT MAIL

การสร้างฟอนต์ไทยในแบบเวคเตอร์ให้แสดงบนจอภาพและพล็อตเตอร์



อันว่ารูปแบบของตัวอักษรที่ปรากฏให้เห็นบนหน้าจอหรือบนกระดาษพิมพ์นั้น เขาเรียกกันว่าฟอนต์ (Font) เมื่อก่อนนั้นนักโปรแกรมคอมพิวเตอร์ก็ไม่ค่อยจะได้สนอกสนใจกับคำว่าฟอนต์เท่าไร บางคนอาจจะไม่รู้จักเสียด้วยซ้ำไป ก็เมื่อก่อนนี้เทคโนโลยีทางด้านคอมพิวเตอร์ยังจำกัดอยู่เฉพาะในเรื่องของการเขียนโปรแกรมจริง ๆ มีน้อยคนที่จะเข้าไปซาบซึ้งกับเทคโนโลยีทางด้านฮาร์ดแวร์ หลายท่านอาจจะสงสัยว่า คำว่า "เมื่อก่อนนี้" มันเมื่อก่อนไหนกันแน่ ก็ไม่นานเท่าไหร่หรอกครับ สัก 10 กว่าปีมานี้เอง พอเครื่องไมโครคอมพิวเตอร์เริ่มเข้ามาก็คงจะราวปี พ.ศ. 2525 ได้กระมังครับ ลักษณะของการเข้ามาของไมโครคอมพิวเตอร์นำเอาเทคโนโลยีมาแพร่หลายเข้ามาด้วยไมโครคอมพิวเตอร์ที่เป็นที่นิยมในสมัยแรก ๆ ก็คงจะเป็นเครื่องแอปเปิ้ล ตามด้วย IBM/PC ทั้งสองตัวนี้เปิดเผยเทคโนโลยีให้เห็นเต็มตัว ใครนึกอยากจะดัดแปลงแก้ไขตรงไหนก็เปลี่ยนแปลงเพิ่มเติมได้ตามใจชอบ ถึงตอนนั้นก็ต้องนึกถึงพระมหากรุณาธิคุณของพ่อขุนรามคำแหง ที่ทำให้เรามีรูปแบบอักขระของเราเอง จุดนี้เป็นจุดที่ทำให้นักวิศวกร นักวิทยาศาสตร์ และนักประดิษฐ์ทั้งหลายลงมือศึกษาหาความรู้ทางด้านคอมพิวเตอร์อย่างจริงจังทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์เพื่อที่จะดัดแปลงให้คอมพิวเตอร์เขียนภาษาไทยให้ได้ (ที่จริง

คำว่าเขียนออกจะผิดความหมายไปบ้าง น่าจะใช้คำว่า พิมพ์ จะเหมาะสมกว่า ผมมีเหตุผลจะพูดกันทีหลัง) ผลพลอยได้จากความพยายามเหล่านั้นทำให้นักประดิษฐ์ชาวไทยได้รู้เทคโนโลยีของคอมพิวเตอร์อย่างแท้จริง ไม่เช่นนั้นก็แก้ไขเปลี่ยนแปลงการพิมพ์ภาษาไทยบนจอไม่ได้

จากความพยายามในการนำเสนอภาษาไทยบนจอและเครื่องพิมพ์ให้ได้ ทำให้เรารู้จักคำว่าฟอนต์ดีขึ้น

เอาละ เราสามารถสร้างฟอนต์ให้แสดงผลบนจอได้ พิมพ์ออกที่เครื่องพิมพ์ชนิดดอตแมตริกซ์ก็ยังพอได้ พอจะนำออกที่พล็อตเตอร์มีปัญหาละซี ไม่สามารถจะใช้รูปแบบอักขระหรือฟอนต์ชุดเดียวกับที่แสดงออกที่หน้าจอได้ เออ! แปลก โธ่คุณเอ๋ยไม่เห็นมันจะแปลกประหลาดมหัศจรรย์ตรงไหนเลย คุณคงนึกอยู่ในใจ ใคร ๆ เขาก็เคาออกว่าเครื่องพิมพ์กับพล็อตเตอร์มันคนละเครื่องกัน ยังจะแสดงความฉลาด มาบอกว่าแปลกอีก เอาละไม่แปลกก็ไม่แปลก มาดูกันดีกว่าว่ามันต่างกันตรงไหน เพื่อว่าเราจะได้หาช่องทางทำฟอนต์สำหรับพล็อตเตอร์ได้

ข้อแตกต่างที่สำคัญคือ วิธีการพิมพ์ลงบนกระดาษเครื่องพิมพ์แบบดอตแมตริกซ์เรียกได้เต็มปากว่า"พิมพ์" เพราะรูปแบบของอักษรถูกกระแทกพิมพ์ลงบนกระดาษเป็นจุด ๆ ตามรูปแบบที่กำหนดให้ทีละคอลัมน์จากซ้ายไปขวา แต่ว่าเครื่องพล็อตเตอร์นั้น เครื่องมือที่สำคัญคือปากกาด้ามเดียว ลากเส้นไปมาจนเกิดเป็นรูปร่างของตัวอักษร จะเรียกว่าพิมพ์ก็คงจะไม่ถนัดนัก เรียกว่าเขียนแหละเหมาะสมที่สุด ดังนั้น กำหนดฟอนต์สำหรับให้เครื่องพิมพ์แบบดอตแมตริกซ์จึงเป็นฟอนต์ที่มีรูปแบบเป็นจุดที่ประกอบกันเป็นรูปตัวอักษร ในขณะที่ฟอนต์ที่จะให้เครื่องพล็อตเตอร์เขียนนั้น เป็นการกำหนดคำสั่งให้เครื่องพล็อตเตอร์ลากเส้นไปในทิศทางและระยะทางต่าง ๆ จนเกิดเป็นลายลักษณ์ของตัวอักษร อย่างนี้ซิ ถึงจะเรียกว่าสอนเครื่องพล็อตเตอร์ให้สามารถเขียนภาษาไทยอย่างแท้จริง

บิตแม็พกับเวคเตอร์

ฟอนต์ที่มีรูปแบบจุดประกอบกันในลักษณะของตาราง 2 มิติ โดยกำหนดจุดในตาราง 2 มิติให้เป็น ขาวหรือดำ เพื่อต่อกันเป็นรูปแบบของอักขระ เรียกว่าฟอนต์แบบบิตแม็พ เพราะการเก็บรูปแบบของอักขระเหล่านี้ทำโดยกำหนดให้ทุกจุดบนตารางเกี่ยวโยงกับหน่วยความจำ 1 บิต ฟอนต์แบบนี้ใช้ในการแสดงผลบนจอซีอาร์ทีที่เราเห็นโดยทั่ว ๆ ไปเช่นกัน รูปลักษณะของอักขระโรมันนั้น สามารถใช้ตาราง 8×8 จุดแทนได้ โดยคุณภาพของอักขระที่เห็นนับว่าอยู่ในเกณฑ์ที่ยอมรับได้ แต่สำหรับภาษาไทยแล้ว 8×8 เห็นท่าจะไม่เพียงพอเสียแล้ว เพราะอักษรไทยมีทั้งหัวทั้งหาง อีกทั้งสระ วรรณยุกต์ ที่เรียงตัวอยู่ข้างบนและข้างล่าง ทำให้ไม่สามารถจะได้คุณภาพที่ยอมรับได้ ไม่เชื่อลองนั่งออกแบบอักษร "ฏ" หรือ "ฒ" บนตาราง 8×8 ดูซิครับ ภาษาไทยบนจอรุ่นแรก จึงอ่านค่อนข้างจะลำบาก ในยุคหลังมีการใช้ความละเอียดของจอ VGA ให้เกิดประโยชน์ จึงมีการสร้างรูปแบบบนตาราง 8×20 คือกว้าง 8 จุด สูง 20 จุด ทำให้การอ่านดูง่ายเข้า สำหรับเครื่องพิมพ์อาจจะสร้างให้มีคุณภาพดีจนถึงขนาด 16×24 จุด ก็ยังได้ เนื่องจากเครื่องพิมพ์ยุคใหม่มีขีดความสามารถในการควบคุมระยะห่างระหว่างคอลัมน์ได้แคบเข้า

ฟอนต์ที่มีลักษณะเป็นคำสั่งในการลากเส้นแบบที่ใช้สั่งเครื่องพล็อตเตอร์นั้นแตกต่างไปจากแบบบิตแม็พโดยสิ้นเชิง การเก็บฟอนต์แบบนี้ ก็คือการเก็บคำสั่งในการลากเส้นของเครื่อง อาจจะเป็นการลากเส้นตรงอย่างเดียวหรือมีคำสั่งอื่น ๆ มาผสมเช่นเส้นโค้ง วงกลม ครึ่งวงกลม หรือส่วนของเส้นโค้ง เพื่อทำให้รูปแบบอักษรออกมาสวยงาม เสมือนหนึ่งว่าเป็นกำหนดทิศทางของการเขียนเอาไว้ จะเรียกว่าฟอนต์แบบเวคเตอร์ก็เห็นจะไม่ผิดตรงไหน เพราะช่วงของการลากเส้นมีทิศทางที่แน่นอนที่จะไปทางไหน

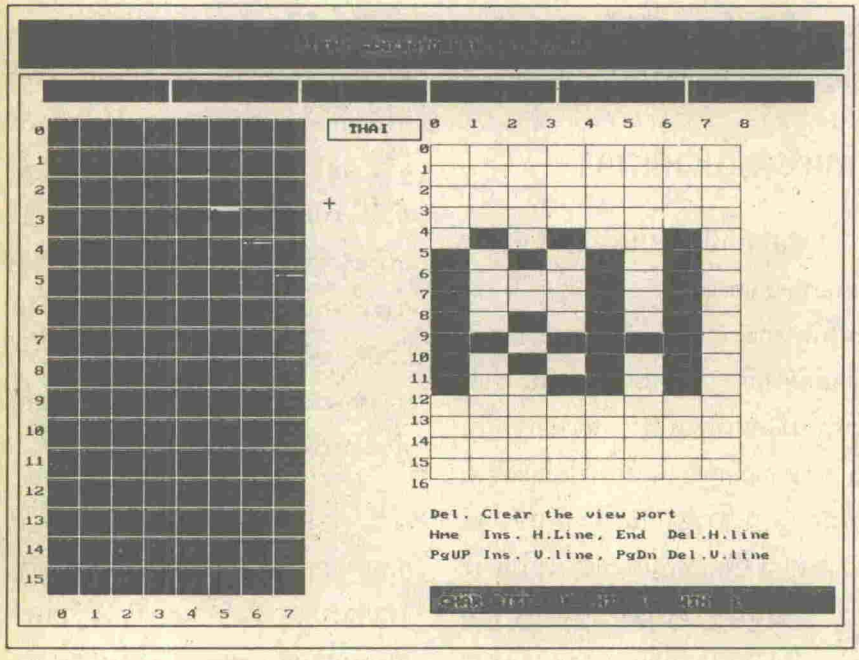
การแสดงผลบนจอแทบทุกรายเป็นการแสดงผลแบบบิตแม็พ เนื่องจากง่ายและแสดงผลได้รวดเร็วกว่าวิธีการใช้ฟอนต์แบบเวคเตอร์ โปรแกรมทางด้านกราฟิกหลายตัวใช้เวคเตอร์ฟอนต์สำหรับการแสดงผล แม้ว่าการแสดงผลจะช้ากว่าการใช้ฟอนต์แบบบิตแม็พก็ตาม สาเหตุของการใช้เวลาในการเขียนบนจอมากกว่าแบบบิตแม็พก็เนื่องจากว่าซีพียูจะต้องเสียเวลาในการแปลงเวคเตอร์ฟอนต์เขียนออกบนจอ ซึ่งเป็นบิตแม็พระหว่างหน่วยความจำแสดงผลกับจุดและจุดบนหน้าจอ กระนั้นก็ตาม การใช้ฟอนต์แบบเวคเตอร์เริ่มเข้ามาแพร่หลายด้วยสาเหตุหลายประการ

- การย่อขยายทำได้โดยง่าย และไม่เสียความละเอียด
- ความละเอียดของการแสดงอักขระไม่ขึ้นอยู่กับเครื่องที่สร้างมันขึ้นมา แต่ขึ้นอยู่กับเครื่องที่มันกำลังแสดงอยู่

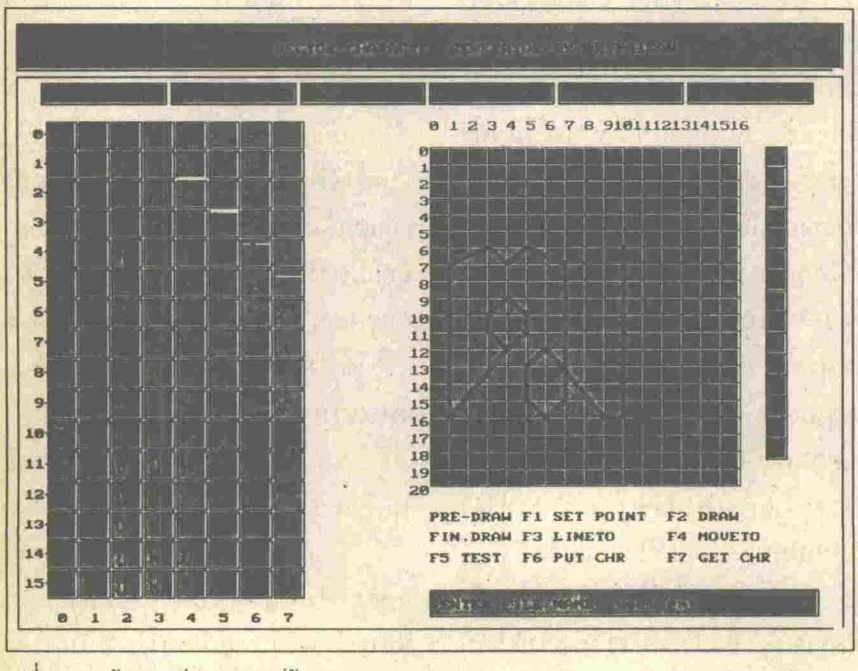
- การหมุนหรือเอียงตัวอักขระทำได้ง่าย

อะดอปโพสต์สคริปต์เป็นผู้นำในการสร้างกราฟิกแบบเวคเตอร์ ภาพกราฟิกทุกภาพเก็บไว้ในลักษณะของคำสั่งในการเขียนภาพพื้นฐาน เช่น ลากเส้น เขียนเส้นโค้ง เขียนวงกลม เป็นต้น ภาพที่ได้จากการนำเอาแฟ้มของโพสต์-สคริปต์เมื่อนำมาสร้างให้เกิดเป็นภาพขึ้นมาใหม่จึงสามารถใช้ความละเอียดสูงสุดของเครื่องที่สร้างมันขึ้นมาใหม่ไม่ได้ขึ้นอยู่กับเครื่องที่สร้างมันขึ้นมา มีโปรแกรมกราฟิกอีกหลายตัวที่ใช้ฟอนต์ในลักษณะเดียวกับโพสต์สคริปต์ เช่น Dr. Halo, Corel Draw เป็นต้น

ลองเปรียบเทียบคุณภาพที่ได้จากฟอนต์ทั้งสองชนิดในรูปที่ 1 และรูปที่ 2



รูปที่ 1 การสร้างฟอนต์แบบบิตแม็พขนาด 8x16



รูปที่ 2 การสร้างฟอนต์แบบเวคเตอร์ด้วยตาราง 16x20

มาลองสร้างฟอนต์แบบเวคเตอร์ดูไหม

ก่อนที่จะสร้างฟอนต์แบบเวคเตอร์ต้องถามตัวเองก่อนว่าจะสร้างเพื่อไปใช้กับอะไร เพราะฟอนต์แบบนี้ต้องเข้าใจกันระหว่างผู้สร้างและผู้นำไปใช้งาน มันไม่เหมือนกับฟอนต์แบบบิตแม็พ ซึ่งมีวิธีการนำไปใช้ที่ตายตัว ไม่มีพลิกแพลงเป็นอื่นไปได้ ที่ว่าจะต้องรู้กันระหว่างการเอาไปใช้งานกับการสร้างขึ้นมาก็เพื่อให้การสร้างตรงตามจุดประสงค์ของผู้ใช้ อย่างเช่น หากเราจะสร้างฟอนต์สำหรับเครื่องที่เข้าใจภาษาโพสต์สคริปต์ เราก็ต้องรู้ว่าโพสต์สคริปต์เขามีคำสั่งอย่างไรบ้างเพื่อจะได้นำมาเข้ารหัสคำสั่งได้ถูกต้อง ถ้าจะสร้างฟอนต์สำหรับออโต้แคดเราก็จะต้องทราบว่าออโต้แคดเขียนอย่างไร (นอกจากจะใช้โปรแกรมของออโต้แคดสร้างเองและใช้เอง) หรือหากจะสร้างฟอนต์สำหรับ ดร.ฮาโล เราก็จะต้องทราบว่า ดร.ฮาโลเข้ารหัสคำสั่งในการเขียนอักษรอย่างไร เพราะต่างคนต่างมีวิธีการที่ไม่เหมือนกันในการเข้ารหัสคำสั่ง ยังไม่มีมาตรฐานที่แน่นอน มีเพียงแต่โพสต์สคริปต์เท่านั้นที่ทำท่าว่าจะเป็นมาตรฐานโดยปริยาย (DeFacto Standard) ฟอนต์พวกนี้ส่วนมากจะนำมาใช้แทนกันไม่ได้เลย ถ้าหากเราไม่เข้าใจหรือยังไม่มีข้อมูลทางเทคนิคก็คงจะทำได้อย่างเดียวคือ

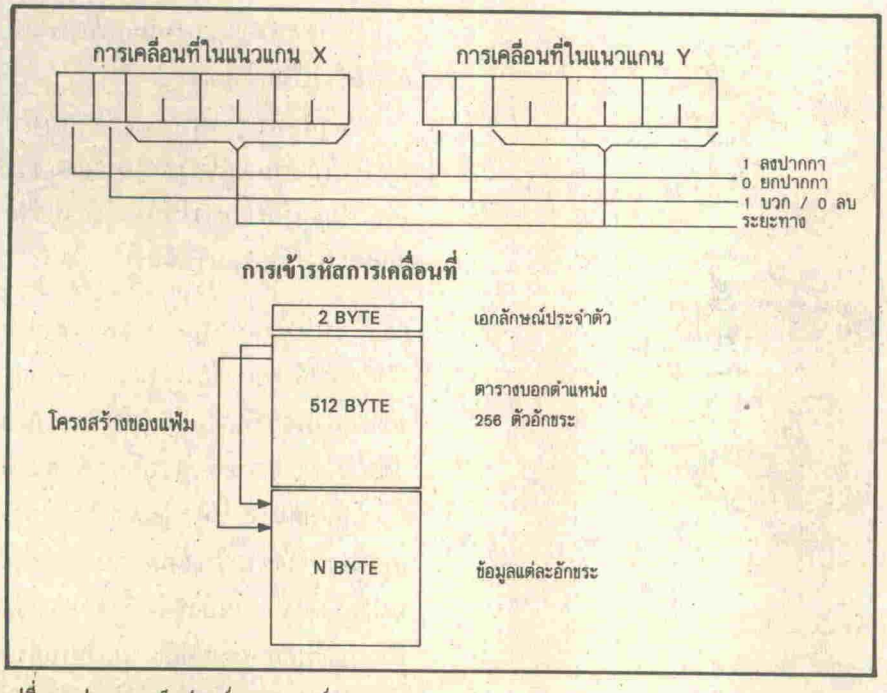
สร้างเองใช้เอง ทำมาตรฐานของตัวเอง ไม่ต้องยุ่งกับคนอื่นเขา ดังที่ผมกำลังจะแสดงให้ดูต่อไปนี้

ผมพยายามหาข้อมูลทางเทคนิคของการเก็บฟอนต์ของโพสต์สคริปต์ และของดร.ฮาโลก็ยังหาข้อมูลทางเทคนิคเหล่านั้นไม่ได้ ลองเอาฟอนต์มานั่งดีบักดูก็จนใจจับต้นชนปลายไม่ถูก ผมก็เลยออกแบบเองนำไปใช้เอง คือเขียนโปรแกรมสร้างฟอนต์เอง เขียนโปรแกรมการใช้งานกราฟิกเองหมดเรื่องหมดราวจะได้รู้เองหมดทุกอย่างมาลองดูซิว่าผมทำอย่างไร ท่านอาจจะมีแนวความคิดที่ดีกว่าผมนำไปสร้างเสริมต่อให้เกิดเป็นการพัฒนาที่ไม่ต้องพึ่งพาเทคโนโลยีที่มาจากต่างประเทศ

ก่อนอื่น ผมสร้างสมมุติฐานขึ้นมาก่อนว่า รูปแบบอักษรที่จะสร้างขึ้นมาประกอบด้วยการลากเส้นตรงจากจุดหนึ่งไปยังอีกจุดหนึ่งเท่านั้น ยังไม่มีการนำส่วนโค้งเข้ามาเกี่ยวข้องเพื่อความสะดวกและง่ายต่อการนำไปใช้งาน แต่ละส่วนของเส้นตรงจะมีการเปลี่ยนแปลงในแนวแกน X และแนวแกน Y ไม่เกิน 64 หน่วย ในครั้งแรกนี้คิดว่าคงจะได้ความละเอียดของอักษรเพียงพอแก่ความต้องการ ดังนั้น ผมสามารถที่จะเข้ารหัสทิศทางการเขียน X และ Y ลงใน 2 ไบต์ โดยแต่ละทิศทางใช้ 1 ไบต์เป็นตัวกำหนด สำหรับบิตที่ 8 นั้นสำหรับเป็นตัวบอกว่าลากเส้นตรงโดยสร้างรอยทางด้วย (ลงปากกา) หรือว่าลากผ่านไปเฉย ๆ โดยไม่มีการเขียน บิตที่ 7 ใช้เป็นตัวบอกทิศทางว่าเป็นบวกหรือลบจากจุดเดิมที่ยืนอยู่โดยวิธีการออฟเซท คือ ใน 7 บิตล่างเลขที่เกิน 64 คือค่าบวกของส่วนที่เกิน พูดง่าย ๆ ก็คือเอา 64 ไปลบออกจากค่าที่ได้จาก 7 บิตล่าง ดูรูปที่ 3 ประกอบ

เอาละเมื่อเราได้วิธีการเข้ารหัสที่แน่นอนแล้ว คราวนี้มาดูว่าเราจะรวมเอาอักขระทั้งหมดเข้าไว้ในแฟ้มรูปแบบอักขระทั้งหมดได้อย่างไร เนื่องจากอักขระแต่ละตัวจะมีจำนวนช่องของการเขียนที่แตกต่างกัน เช่น "ก" อาจจะใช้แค่ 6 ช่อง "ฒ" อาจจะต้องใช้ถึง 20 ช่อง (ตามตัวอย่างในรูปที่ 2) ดังนั้นการมองหาตำแหน่งของฟอนต์จึงไม่แน่นอนเหมือนกับฟอนต์แบบบิตแม็พ ซึ่งอักขระแต่ละตัวใช้หน่วยความจำสำหรับเก็บฟอนต์เท่า ๆ กัน หนทางในการทำก็คือการสร้างตารางสำหรับเป็นพอยน์เตอร์ไว้ที่หัวของแฟ้ม ตารางในส่วนหัวนี้ก็จะประกอบไปด้วย 256 พอยน์เตอร์ สำหรับชี้ไปยังตำแหน่งต่าง ๆ ในแฟ้ม ซึ่งเป็นจุดเริ่มต้นของการเขียนตัวอักษร สำหรับจุดสุดท้ายนั้นใช้ 0 หรือ Null เป็นตัวบอก เนื่องจากเราใช้ 7 บิต ในการบอกระยะทางเป็นแบบออฟเซท 64 นั้นคือค่าที่เกิน 64 ถือว่าเป็นบวก เคลื่อนที่เป็นระยะทาง 0 รหัสที่จะใส่ก็คือ 64 ดังนั้น ค่า 0 ในรหัสเคลื่อนที่ รหัสจึงเปรียบเสมือนกับคำสั่งให้เคลื่อนที่เทียบกับจุดเดิมไป -64 หน่วย ซึ่งเราจะไม่ใช้เพราะตั้งใจว่าใช้ไม่ให้เกินนั้น

ตัวอย่างในรูปที่ 2 เป็นการใช้โปรแกรมสำหรับการสร้างฟอนต์แบบเวคเตอร์ตามวิธีการที่กล่าวข้างบน จะสังเกตเห็นว่า อักษรที่ใช้จะเขียนอยู่บนตาราง 16×20 เท่านั้น ยังไม่ได้ขยายให้เต็มขีดความสามารถ ถึง 64×64 อย่างไรก็ตาม รูปแบบของอักขระที่ได้มีคุณภาพที่ยอมรับได้แม้ว่าจะขยายหรือย่อส่วน



รูปที่ 3 รูปแบบการเก็บฟอนต์แบบเวคเตอร์

โปรแกรมกราฟิกกับการใช้งานฟอนต์

ด้วยความขัดเคืองในอารมณ์ในการใช้โปรแกรมกราฟิกสำหรับใช้เขียนภาษาไทยได้ทั้งในจอภาพและบนพล็อต-

เตอร์จึงลงมือเขียนมันเสียเอง จะได้ใช้ฟอนต์ที่ลองเขียนเอาไว้เองด้วย แม้ว่าจะมีการพัฒนานำเอาภาษาไทยใส่ไปในโปรแกรมกราฟิกได้หลายตัวแล้วก็ตาม แต่ลองดูแนวความคิดของผมบ้างคงจะไม่เสียหลายหรอกนะ

เมื่อแรกเริ่มได้ระบบ VGA มาใหม่ ๆ ได้ลองใช้คร.ฮาโลแล้วรู้สึกติดใจการเขียนอักขระลงบนรูปกราฟิก เพราะสามารถกำหนดให้มันมีขนาดเท่าไรก็ได้บนจอเลยทีเดียว ผมก็เลยดึงเอาส่วนดีตรงนี้มาพัฒนาดูเล่น ๆ อีกทั้งพยายามให้ภาพที่ได้จากการเขียนบนจอเก็บเอาไว้ในลักษณะของคำสั่ง เพื่อที่จะได้สามารถนำไปสร้างใหม่ด้วยขนาดที่เปลี่ยนแปลงได้ตามความต้องการ โปรแกรมกราฟิกที่ได้จึงเป็นไปในลักษณะของลายเส้น โดยมีคำสั่งพื้นฐานจริง ๆ คือ การลงจุดบนหน้าจอ จากคำสั่งพื้นฐานนี้ก็นำมาสร้างคำสั่งที่สูงขึ้นมาคือ

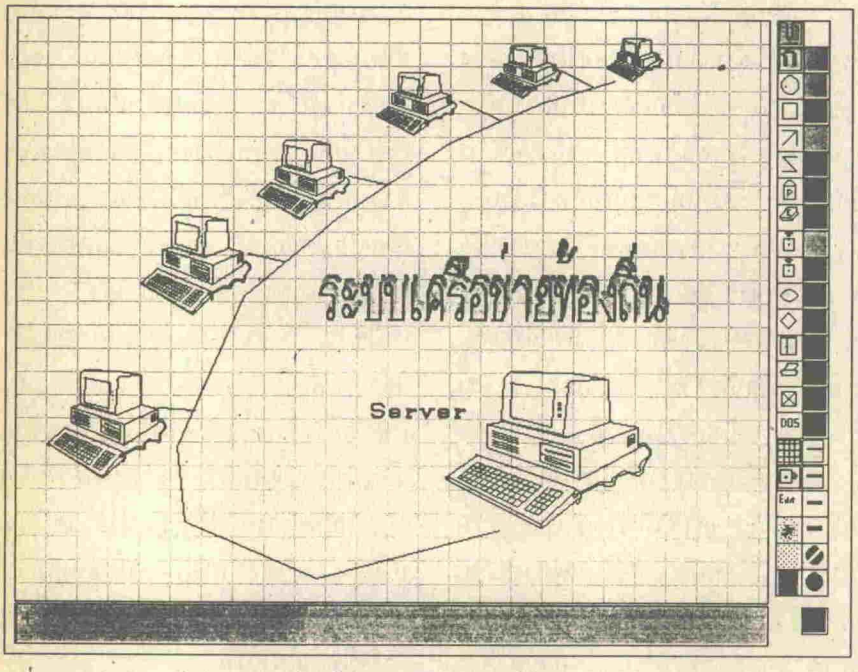
- การลากเส้น
- การเขียนวงกลม และ Ellipse
- การเขียนส่วนของวงกลม
- การเขียนสี่เหลี่ยม
- การเขียนสี่เหลี่ยมแหลม
- การเลือกสี
- การเปลี่ยนความหนาของลายเส้น

ในส่วนของการสนับสนุนการสร้างภาพก็มี

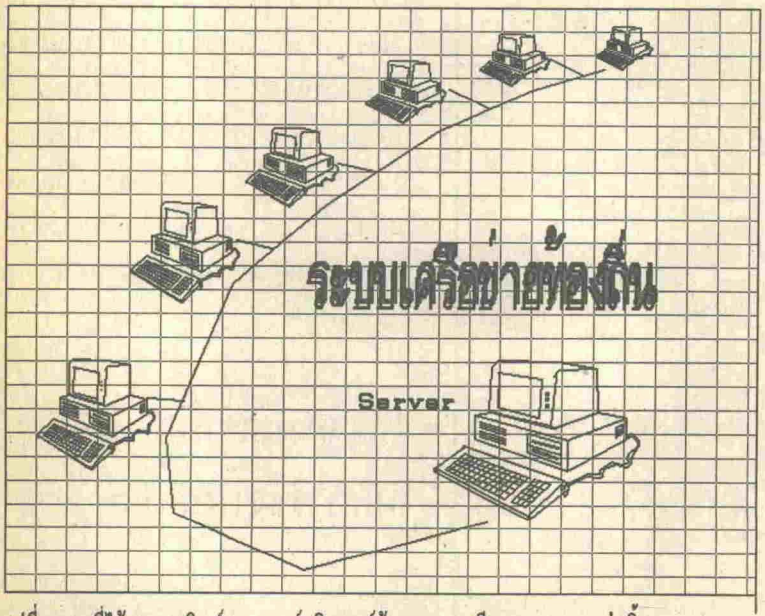
- การเขียนตัวอักษรในลักษณะของสายวลี
- การอ่านแฟ้มภาพเก่าออกมาแสดงในลักษณะของการย่อหรือขยายลงในตำแหน่งต่าง ๆ บนภาพ
- การเก็บแฟ้มภาพลงดิสค์
- การสั่งเขียนภาพบนเครื่องพิมพ์เลเซอร์ หรือพล็อตเตอร์

คำสั่งพื้นฐานเหล่านี้ปรากฏบนหน้าจอในลักษณะของเมนูทางด้านขวามือ เมาส์มีส่วนทำให้การใช้งานง่ายและสะดวกสบายมากขึ้น

ในที่นี้จะขอพูดถึงเฉพาะส่วนที่เกี่ยวข้องกับการเขียนสายวลี (String) ลงบนภาพกราฟิกมาคุยสู่กันฟังเท่านั้น วิธีการใช้งานสำหรับการเขียนข้อความก็คือ เลื่อนเมาส์ไปที่สัญญกร 'ก' โปรแกรมจะให้ท่านใส่ข้อความผ่านทางหน้าต่างด้านล่างของจอ ทั้งไทย หรืออังกฤษ ในลักษณะเช่นเดียวกับโปรแกรมเวิร์ดโปรเซสเซอร์ทั่ว ๆ ไป เพียงแต่



รูปที่ 4 ตัวอย่างโปรแกรมการนำฟอนต์ไปใช้เขียนภาพในแบบเวคเตอร์



รูปที่ 5 ผลที่ได้จากการพิมพ์บนเลเซอร์พรินเตอร์ด้วยความละเอียด 150 จุดต่อนิ้ว ขนาดของภาพ 1280x960 จุด 1 พิกเซลแทนด้วย 2x2 จุด

จำกัดไว้ครั้งละไม่เกิน 60 ตัวอักษร เท่านั้น เมื่อกดแป้น 'Enter' จะปรากฏกล่องสี่เหลี่ยมบนจอ กล่องสี่เหลี่ยมนี้ก็คือตำแหน่งและขนาดของข้อความที่จะปรากฏลงบนภาพ ท่านสามารถเลื่อนกล่องสี่เหลี่ยมไปไว้ตรงไหนก็ได้ โดยการเลื่อนเมาส์ หากจะขยายหรือย่อส่วนไม่ว่าจะทางแกน x หรือ y ก็สามารถทำได้โดยการกดปุ่มด้านขวาของเมาส์แล้วเลื่อนเมาส์ ขนาดของกล่องสี่เหลี่ยมก็จะย่อหรือขยายตามทิศทางของการเลื่อนเมาส์ และปุ่มซ้ายมือของเมาส์เป็นการบอกให้เขียนข้อความลงบนภาพได้ อาจจะเขียนได้ในหลายตำแหน่งในสำหรับการใส่ข้อความ 1 ครั้งจนกว่าจะกดปุ่ม Esc ลักษณะการใส่ข้อความเช่นนี้ทำให้ง่ายต่อการสร้างภาพหรือสร้างชาร์ตแทนที่จะเป็นการเลือกดึงเอาจากตาราง ดังเช่นในออโต้แคดหรือโปรแกรมอื่น ๆ ซับรูทีนที่สำคัญในการเขียนด้วยระบบหน้าจอแสดงอยู่ในโปรแกรมรูปที่ 8

จากตัวอย่างของโปรแกรมซึ่งเขียนโดยภาษาซี จะพบว่าคำสั่งพื้นฐานก็คือการลากเส้นและการเคลื่อนตำแหน่งไปโดยไม่มีการลากเส้นไปยังตำแหน่งใหม่สัมพัทธ์กับจุดเดิม คือ

lineto()

Moveto()

หากรูทีนทั้งสองนี้เขียนโดยภาษาเครื่องก็จะทำให้การทำงานของโปรแกรมเร็วเข้า

สำหรับการส่งภาพออกที่เครื่องพล็อตเตอร์ สามารถทำได้ง่ายเข้าโดยการเปลี่ยนรูทีน lineto() และ moveto () ให้เป็นคำสั่งง่าย ๆ ของเครื่องพล็อตเตอร์ เช่น

```
lineto(x,y)
{
    fprintf(prn,"PD;PA;%d,%d",
x,y);
    return(0);
    }
    moveto(x,y)
    {
    fprintf(prn,"PU;PA;%d,%d",
x,y);
    return(0);
    }
```

PU คือคำสั่งยกปากกาขึ้น

PD คือคำสั่งเอาปากกาลง

PA คือคำสั่งเลื่อนไปยังตำแหน่งสัมบูรณ์ ที่ X, Y ที่ตามมา

ลูกเล่นของการเขียนตัวหนังสือ

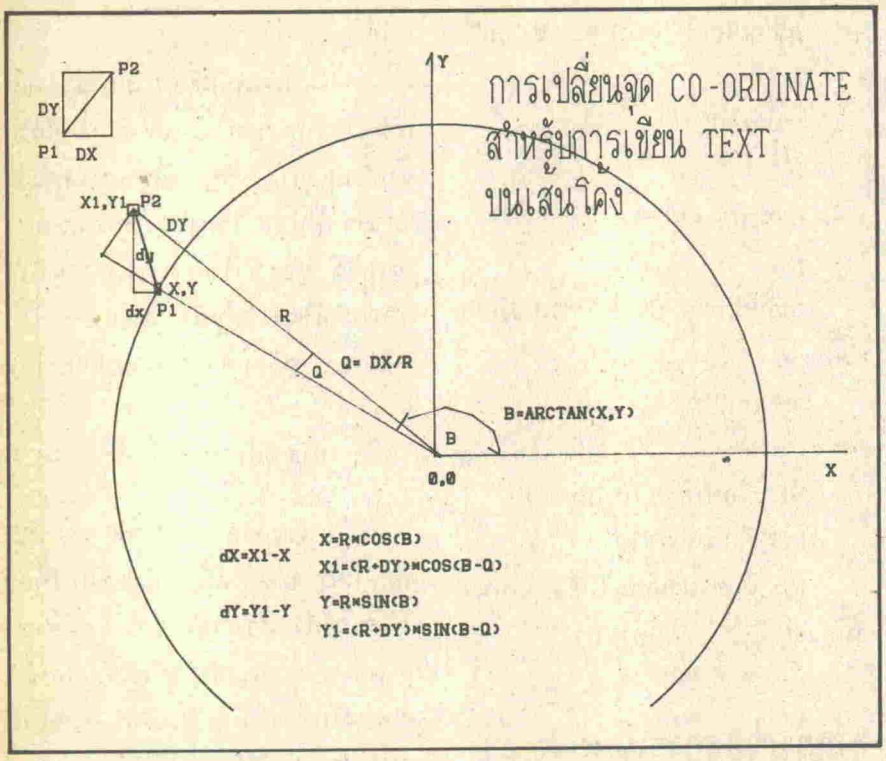
การบรรยายวิธีการเขียนอักขระให้แก่เครื่อง แทนที่จะเป็นรูปแบบตาราง ดังเช่นฟอนต์แบบบิตแม็พจะทำให้เราสามารถทำลูกเล่นกับตัวหนังสือได้หลายแบบ เช่น

- การเขียนเปลี่ยนทิศทาง จากเดิมซ้ายไปขวา สามารถกำหนดให้เป็นจากบนลงล่าง จากล่างขึ้นบน จากขวาไปซ้าย เป็นต้น วิธีการทางคณิตศาสตร์เข้ามาช่วยจะทำให้การเขียนดังกล่าวเป็นไปได้โดยใช้ฟอนต์ชุดเดิม
- การเขียนตัวเอียงโดยมุมเท่าไหร่ก็ได้
- การเขียนบนส่วนโค้ง

ในบรรดาลูกเล่นที่กล่าวข้างบนที่ค่อนข้างจะยากก็คือ การเขียนให้อยู่ในส่วนโค้ง ส่วนวิธีการเปลี่ยนทิศทางการเขียน และการทำตัวเอียงนั้นค่อนข้างจะตรงไปตรงมา เพราะเป็นการสลับค่ากันระหว่าง x กับ y และ −x, −y ส่วนการเขียนแบบตัวเอียงนั้นต้องอาศัยหลักการของมุมเอียงมาช่วย มีการย้ายแกนโคออร์ดิเนตไปเทียบกับจุดเริ่มต้นของการเขียนตัวอักขระ ข้อสำคัญที่พึงคำนึงถึงก็คือ จะต้องหาระยะทางจากจุดเริ่มของตัวอักขระให้ได้เสียก่อนที่จะ

รูปที่ 6 ตัวอย่างการใช้ฟอนต์เขียนหนังสือในหลายทิศทาง และการทำตัวเอียง

FONT (THAI)
การสร้างฟอนต์เพื่อกราฟิกแบบเวคเตอร์



รูปที่ 7 แนวทางการเขียนอักขระบนส่วนโค้ง

ทำการย้ายแกน เนื่องจากการบรรยายลักษณะการเขียนอักขระของเราเป็นแบบสัมพัทธ์กับจุดสุดท้าย ลองดูตัวอย่างในโปรแกรมรูปที่ 9 สำหรับการเปลี่ยนทิศทางการพิมพ์และการทำตัวเอียง พอเป็นตัวอย่างเพื่อความเข้าใจที่ดีขึ้น แม้ว่าจะยังไม่ถูกต้องตามหลักการย้ายแกนร้อยเปอร์เซ็นต์แต่ก็ให้ผลไม่แตกต่างกันมากนัก ข้อสำคัญคือทำให้มันทำงานเร็วกว่าวิธีการนำเอาการย้ายแกนแบบโพลาร์เข้ามาช่วยเพราะไม่ต้องคำนวณมาก

รูปที่ 6 เป็นตัวอย่างการใช้รูทีนดังกล่าวเขียนหนังสือบนจอในแบบต่าง ๆ โดยใช้ฟอนต์ตัวเดิม

ในส่วนของการเขียนอักขระบนส่วนโค้งนั้น ผมยังไม่ได้ลองทำแต่คิดว่าคงไม่ยากเย็นอะไร โดยขอเสนอแนวความคิดดังนี้

สมมุติว่าเราต้องการเขียนข้อความ "คอมพิวเตอร์กราฟฟิก" ลงบนส่วนโค้งที่มีรัศมี R จำนวนบล็อกของตัวอักขระเท่ากับ 15 เริ่มตั้งแต่มุม 145 องศา ไปจนถึงมุม 45 องศา ใน 1 บล็อกตัวอักขระจะใช้มุม 90/15 องศา เนื่องจากเรากำหนดให้รูปแบบอักขระเป็นการเขียนเทียบกับจุดก่อนหน้านี้ (ดูรูปที่ 7 ประกอบ)

ถ้า P1 = จุดก่อนหน้านี้ที่เขียน
P2 = จุดใหม่ที่จะเคลื่อนที่ไป
R = รัศมีของจุดก่อนหน้านี้
B = มุมของจุดก่อนหน้านี้ที่เขียน
Q = มุมที่เปลี่ยนแปลงไปยังจุดใหม่
DY = ระยะทางการเคลื่อนที่ทางแกน Y ในระบบปกติ
DX = ระยะทางการเคลื่อนที่ทางแกน X ในระบบปกติ

เราสามารถหาระยะทาง สัมพัทธ์กับจุดเดิมคือ dx และ dy ได้ดังนี้

รูปที่ 8 รูทีนสำหรับเขียนอักขระบนจอภาพ

```
/*************************
  Draw a vector font
*************************/
#define abs(x) ((x)<0?-(x):(x))
int cptr[256];          /*pointer for character gen*/
char cvect[5000];       /*vector character gen*/
int  f_ptr;
extern int setx,        /*location where the x point is set*/
           sety,        /*location where y point is set*/
           lthick;      /*thickness of the line*/

/*****************************
 draw the vector chracter
 *****************************/
draw_c(c,sx,sy)
double sx,sy;           /*size to draw */
char c;               /*description of char to draw*/
 {
  int rx,ry,i,ox,oy,b;
  char *s;
  int px,py,ix,iy,y;


  i=cptr[c];
  s=&cvect[i];
  i=0;
  ox=setx;
  oy=sety;
  b=0;
  if(c==0xd8||c==0xd9)      /*upper the normal line*/
```

รูปที่ 8 (ต่อ)

```
31:        {
32:         b=1;
33:         setx=setx-10*sx;
34:         sety=sety+2*sy;
35:         }
36:      if((c>=0xd4&&c<=0xd7)||(c>=0xe7&&c<=0xed)||c==0xd1)
37:         {                        /*lower the normal line*/
38:          b=1;
39:          setx=setx-11*sx;
40:          sety=sety-12*sy;
41:          }
42:      px=0;
43:      py=0;
44:      ix=setx;iy=sety;
45:      while(*s!=0)
46:       {
47:        i=*(s++);
48:        px=px+(i&0x7f)-64;
49:        y=*(s++);
50:        py=py+(y&0x7f)-64;
51:
52:        rx=px*sx;
53:        ry=py*sy;
54:        if(i>=128)moveto(ix+rx,iy+ry);
55:        else lineto(ix+rx,iy+ry,lthick);
56:        }
57:      if(b!=0){setx=ox;sety=oy;}
58:      else    setx=setx+1.5*sx;
59:      return(0);
60:      }
61:
62:   /********************
63:    display text
64:    ********************/
65:   dtext(s,x1,y1,x,y)
66:   int  x,y,x1,y1;      /*square box where the text to be put there*/
67:   char *s;
68:    {
69:     int l;
70:     double sx,sy;      /*scale factor*/
71:
72:     l=str_width(s);
73:     sx=abs(x1-x);sx=sx/l/10;
74:     sy=abs(y1-y);sy=sy/16;
75:     moveto(x,y1);
76:     while(*s!=0)
77:          {
78:           draw_c(*s,sx,sy);
79:           s++;
80:          }
81:      return(0);
82:      }
83:
84:    /********************
85:    load vector character gen
86:    ********************/
87:   loadvfnt(fnt)
88:   char *fnt;
89:    {
```

รูปที่ 8 (ต่อ)

```
90:    int key,i,cgf;
91:
92:    cgf=dopen(fnt,0);
93:      if(cgf!=-1)
94:       {
95:         dread(cgf,&cvect[0],8);
96:          f_ptr=cvect[2]*256+cvect[3];
97:          dread(cgf,&cptr[0],512);            /*read pointer*/
98:          f_ptr=dread(cgf,&cvect[0],f_ptr);   /*read character descriptor*/
99:         dclose(cgf);
100:       }
101:     else return(-1);
102:     return(0);
103:   }
104: /*****************************
105:  find the true string width for text
106:  *****************************/
107: str_width(s)
108: char *s;
109:  {
110:  int i;
111:  char c;
112:
113:   i=0;
114:   while((c=*(s++))!=0)
115:   if(!(c==0xd8||c==0xd9||(c>=0xd4&&c<=0xd7)||(c>=0xe8&&c<=0xed)||c==0xd1))i++;
116:   return(i);
117:   }
```

รูปที่ 9
รูทีนสำหรับเขียนอักขระที่รวมเอามุมเอียงและทิศทางด้วย

```
1:  /************************************
2:    Draw a vector font with slant angle
3:  ************************************/
4:  #define abs(x) ((x)<0?-(x):(x))
5:  int cptr[256];          /*pointer for character gen*/
6:  char cvect[5000];       /*vector character gen*/
7:  int  f_ptr;
8:  double sin(),cos(),sqrt();
9:  extern int setx,        /*location where the x point is set*/
10:            sety,        /*location where y point is set*/
11:            lthick;      /*thickness of the line*/
12:
13: double s_angle=0;         /*slant angle relative to vertical line,
14:                             in degree */
15: double d_angle=0;         /*direction angle counter clockwise*/
16:
17: /***************************************
18:  draw the vector chracter with slant angle
19:  ***************************************/
20: draw_c(c,sx,sy)
21: double sx,sy;       /* size to draw */
22: char c;             /* description of char to draw*/
23:  {
24:   int rx,ry,i,ox,oy,b;
25:   char *s;
26:   int px,py,ix,iy,y;
27:   double r,ax,ay,bx,by;
28:
29:   i=cptr[c];
30:   s=&cvect[i];
```

รูปที่ 9 (ต่อ)

```
31:     i=0;
32:     ox=setx;                /*where to start from*/
33:     oy=sety;
34:     b=0;
35:
36:     if(c==0xd8||c==0xd9)      /*upper the normal line*/
37:        {                      /*go back to last block at upper point*/
38:         b=1;
39:         bx=10*sx+2*sy*sin(s_angle);
40:         by=2*sy;
41:
42:         setx=setx-bx+by*sin(d_angle);
43:         sety=sety+by+bx*sin(d_angle);
44:         }
45:     if((c>=0xd4&&c<=0xd7)||(c>=0xe7&&c<=0xed)||c==0xd1)
46:        {                      /*lower the normal line*/
47:         b=1;                  /*go back to last block at lower point*/
48:
49:         bx=11*sx+12*sy*sin(s_angle);
50:         by=12*sy;
51:
52:         setx=setx-(bx-by*sin(d_angle));
53:         sety=sety-(by+bx*sin(d_angle));
54:         }
55:
56:     px=0;
57:     py=0;
58:     ix=setx;iy=sety;          /*the beginning point of the character*/
59:     /*draw the character until the end of description*/
60:     while(*s!=0)
61:      {
62:       i=*(s++);
63:       px=px+(i&0x7f)-64;
64:       y=*(s++);
65:       py=py+(y&0x7f)-64;
66:
67:       /***new point to go to */
68:       bx=px*sx;
69:       by=py*sy;
70:
71:       /*add the slant angle*/
72:       ax=bx+by*sin(s_angle);
73:       ay=by;
74:
75:       /**add the direction angle*/
76:       bx=ax-ay*sin(d_angle);
77:       by=ay+ax*sin(d_angle);
78:
79:       rx=bx;
80:       ry=by;
81:       if(i>=128)moveto(ix+rx,iy+ry);     /*move to absolute location*/
82:       else lineto(ix+rx,iy+ry,lthick);   /*draw a line to absolute loc.*/
83:       }
84:
85:     if(b!=0){setx=ox;sety=oy;}      /*set point back to origin if
86:                                       not a normal line*/
87:     else    setx=setx+1.5*sx;
88:     return(0);
89:     }
```

$$
\begin{aligned}
dx &= X1 - X \\
&= (R + DY) * \cos(B - Q) - R * \cos(B) \\
dy &= Y1 - Y \\
&= (R + DY) * \sin(B - Q) - R * \sin(B)
\end{aligned}
$$

ทั้งนี้เราถือว่าจุด 0, 0 อยู่ที่จุดศูนย์กลางของวงกลม เพื่อให้การเขียนเป็นไปอย่างถูกต้องต้องไม่ลืมเอาจุดออฟเซทของจุดศูนย์กลางมาหักออกไปด้วย และแกน Y บนหน้าจอมีทิศทางตรงกันข้ามกับระบบ x,y ทั่วไป กล่าวคือ แกน Y มีทิศทางเป็นบวกเหมือน y เลื่อนจากบนลงล่าง

หนทางการพัฒนาต่อ

อักขระไทยนั้นมีศิลปะในตัวมันเอง มีการตกแต่งและดัดแปลงโฉมได้หลายรูปแบบ การใช้เส้นตรงอย่างเดียวสำหรับวิธีการเขียนคงไม่เพียงพอที่จะทำให้ตัวอักขระสวยงามได้ หลายรูปแบบมีส่วนโค้ง และวงกลมเข้ามาเกี่ยวข้องด้วย อย่างไรก็ตามอักขระไทยมีหลายส่วนที่ซ้ำ ๆ กันเช่นส่วนหัวของอักขระ ซึ่งอาจจะมีสองสามรูปแบบคือ หัวคว่ำขวา หัวคว่ำซ้าย หัวออก หัวเข้าฯ ในส่วนอื่น ๆ ก็เช่นกัน ยกตัวอย่างเช่น ขมวดใน โค้งหลังฯ เป็นต้น หากเราสามารถกำหนดกลุ่มของคำสั่งสำหรับสร้างลักษณะดังกล่าว แล้วแทนนำกลุ่มของคำสั่งดังกล่าวด้วยรหัสตัวเดียวในการบรรยายวิธีการสร้างอักขระ เราก็จะได้ฟอนต์ที่มีขนาดสั้นลงแต่มีประสิทธิภาพสูง อย่างนี้เรียกว่าเรากำลังทำเมตาฟอนต์เข้าไปแล้ว ลองช่วยกันนึก ช่วยกันออกแบบดูซิครับ ผมว่ามันก็สนุกดีเหมือนกันนา เป็นการเอาศิลป์มาช่วยคลายเครียดด้วย